

GESTIONE DELLE IMMAGINI

Il PHP offre anche la possibilità di gestire le immagini, come ad esempio sapere la dimensione di un'immagine JPEG, GIF, PNG, SWF, TIFF e JPEG2000, attraverso l'uso di funzioni specifiche; inoltre, se avete installato le **librerie GD** (scaricabili su <http://www.boutell.com/gd/>) sarete anche in grado di creare immagini al volo e di modificarle.

Funzioni predefinite

Di seguito sono riportate tutte le funzioni di php relative alla gestione di immagini, che non richiedono l'installazione della libreria GD:

Funzione	Esempio	Descrizione
getimagesize	<code>print_r(getimagesize("img.jpg"));</code>	<p>Tale funzione serve per ottenere un array contenente tutte le informazioni relative all'immagine (di qualsiasi estensione, GIF, JPG, PNG, SWF, SWC, PSD, TIFF, BMP, IFF, JP2, JPX, JB2, JPC, XBM, o WBM) passata come parametro, tra cui la larghezza, altezza, formato immagine (MIME TYPE).</p> <p>L'output sarà:</p> <pre>Array ([0] => 337 [1] => 500 [2] => 2 [3] => width="337" height="500" [bits] => 8 [channels] => 3 [mime] => image/jpeg)</pre>

Immagini con GD library

Per installare la libreria GD su Windows (per l'installazione su Linux/Unix leggere la guida ufficiale), basta aprire il file di configurazione php.ini, andare alla linea 550 ed eliminare il ";" davanti all'estensione gd2 (**extension = php_gd2.dll**). In questo modo, riavviando Apache, il vostro server web dovrebbe essere in grado di interpretare il codice php relativo alle librerie GD.

Notare come sia stata decommentata la libreria GD2 e non la GD, poichè la prima dll è già presente nel pacchetto di php 4 e superiori, mentre la seconda no.

Per avere una ulteriore conferma dell'avvenuta installazione, basta eseguire lo script **phpinfo()** e cercare la voce GD, e vedere se le relative voci a GIF, JPG e PNG sono marcate come "enabled":

gd

GD Support	enabled
GD Version	bundled (2.0.15 compatible)
FreeType Support	enabled
FreeType Linkage	with freetype
GIF Read Support	enabled
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled
XBM Support	enabled

Vediamo ora le principali funzioni ed esempi di questa libreria:

Funzione	Sintassi	Descrizione
imgcreate	<code>imageCreate(\$x, \$y)</code>	Prende solo due parametri, le dimensioni x e y dell'immagine espresse in pixel. Con essa è possibile usare <u>solo 256 tonalità di colore diverse</u> , inoltre non è possibile utilizzare il canale Alpha (la trasparenza). Quando si cercherà di allocare un nuovo colore in realtà si allocherà un colore il più vicino possibile a quello desiderato, ma non esattamente quello voluto! Con questa funzione, il <u>primo colore allocato sarà anche il colore di sfondo dell'immagine</u> .
imageCreateTrueColor	<code>imageCreateTrueColor(\$x, \$y)</code>	Il funzionamento è analogo a quello di <code>imageCreate()</code> solo che con questa funzione è possibile utilizzare tutte le tonalità dello standard RGB, vale a dire <u>16.581.375 colori!</u> Questa funzione è stata introdotta nelle GD 2.0 quindi non è possibile utilizzarla con versioni precedenti. Con questa funzione è possibile utilizzare colori dotati <u>anche dell'alpha channel</u> . C'è anche un'altra differenza rispetto ad <code>imageCreate()</code> , vale a dire che il <u>colore di sfondo qui è sempre nero</u> .
imageColorAllocate	<code>imageColorAllocate(\$imageRef, \$red, \$blue, \$green)</code>	Questa funzione alloca nella palette dell'immagine <code>\$imageRef</code> il colore RGB espresso dalla combinazione dei 3 colori fondamentali (rosso, verde, blu) e ne restituisce l'indice. Nota: i colori possono essere espressi da un <u>numero compreso tra 0 e 255</u> , valori fuori da questo range genereranno un errore
imagearc	<code>imagearc(\$imageRef, \$cx, \$cy, \$w, \$h, \$s, \$e, \$color)</code>	Disegna una ellisse / cerchio di centro (<code>\$cx,\$cy</code>), di larghezza <code>\$w</code> ed altezza <code>\$h</code> ; il punto di partenza nel disegnare l'ellisse, di colore <code>\$color</code> , è rappresentato dai gradi <code>\$s</code> e <code>\$e</code> . Un esempio è mostrato qui .
imageRectangle	<code>imageRectangle (\$imageRef, \$x1, \$y1, \$x2, \$y2, \$color)</code>	Disegna un rettangolo nell'immagine <code>\$imageRef</code> partendo da due punti, il vertice superiore sinistro (<code>\$x1, \$y1</code>) e il vertice inferiore destro (<code>\$x2, \$y2</code>) del rettangolo con il colore <code>\$color</code> . Questa funzione disegnerà solo il bordo del rettangolo lasciandone vuoto il contenuto. Il bordo di questo rettangolo sarà

		di 1 px. Un esempio è mostrato qui . N.B.: ricordatevi che le due coordinate che specificate devono sempre rientrare in quelle di <code>imagecreate</code> .
<code>imagechar</code>	<code>imagechar(\$imageRef, \$font, \$x, \$y, \$string, \$color)</code>	Disegna la prima lettera di una stringa, <code>\$string</code> , del colore <code>\$color</code> , del font specificato (<code>\$font</code> è un intero da 1 a 5) e la posiziona alle coordinate <code>\$x</code> e <code>\$y</code> . Un esempio è mostrato qui .
<code>imagecharup</code>	<code>imagecharup(\$imageRef, \$font, \$x, \$y, \$string, \$color)</code>	Uguale al precedente, solo che disegna la lettera verticalmente. Un esempio è mostrato qui .
<code>imagefill</code>	<code>imageFill(\$imageRef, \$x, \$y, \$color)</code>	Questa funzione corrisponde al secchiello del software Paint di Windows; facendo click nel punto (<code>\$x</code> , <code>\$y</code>) riempiremo del colore <code>\$color</code> la porzione di immagine che racchiude quel punto. Un esempio è mostrato qui .
<code>imagefilledpolygon</code>	<code>imagefilledpolygon(\$imageRef, \$array_points, \$num_points, \$color)</code>	Crea un poligono ripieno del colore <code>\$color</code> nell'immagine specificata. Le coordinate dei vertici vengono specificati nell'array <code>\$array_points</code> , mentre il numero dei vertici in <code>\$num_points</code> (ie. <code>point[0] = x0</code> , <code>point[1] = y0</code> , <code>point[2] = x1</code> , <code>point[3] = y1</code> , ...); per cui per un poligono di 6 vertici, l'array avrà una dimensione di 12 (2 coordinate per vertice). Un esempio è mostrato qui .
<code>imagefilledrectangle</code>	<code>imageFilledRectangle (\$imageRef, \$x1, \$y1, \$x2, \$y2, \$color)</code>	Funziona in modo analogo alla funzione <code>imageRectangle</code> , solo che disegnerà il riempimento del rettangolo. Esempi sono mostrati qui e qui .
<code>imageline</code>	<code>imageLine(\$imageRef, \$x1, \$y1, \$x2, \$y2, \$color)</code>	Disegna una linea nell'immagine <code>\$imageRef</code> unendo i punti (<code>\$x1</code> , <code>\$y1</code>) e (<code>\$x2</code> , <code>\$y2</code>) di colore <code>\$color</code> ; un esempio è mostrato qui .
<code>imagestring</code>	<code>imageString(\$imageRef, \$font, \$x, \$y, \$text, \$color)</code>	Inserisce all'interno dell'immagine <code>\$imageRef</code> il testo <code>\$text</code> alle coordinate del punto (<code>\$x</code> , <code>\$y</code>) utilizzando il font <code>\$font</code> e il colore <code>\$color</code> . Per quanto riguarda il font bisogna citare il fatto che sono messi a disposizione 5 font inclusi nelle librerie, è quindi possibile specificare un numero intero compreso tra 1 e 5 per usare quelli interni, oppure specificare il path di un font residente sul disco del server. Un esempio è mostrato qui .
<code>imagestringup</code>	<code>imageStringUp(\$imageRef, \$font, \$x, \$y, \$text, \$color)</code>	Uguale al precedente solo che la stringa è scritta verticalmente.
<code>imagecopyresized</code>	<code>imageCopyResized(\$imgDest, \$imgSource, \$destX, \$destY, \$srcX, \$srcY, \$dstW, \$dstH, \$srcW, \$srcH)</code>	La classica funzione tanto usata per creare le thumbnail . Copia il rettangolo dell'immagine <code>\$imgSource</code> che ha come vertice superiore sinistro il punto (<code>\$srcX</code> , <code>\$srcY</code>) ha come larghezza <code>\$srcW</code> e altezza <code>\$srcH</code> nell'immagine <code>\$imageDest</code> , più precisamente nella porzione delimitata dal rettangolo avente come vertice superiore sinistro il punto (<code>\$destX</code> , <code>\$destY</code>), ha come altezza <code>\$dstH</code> e larghezza <code>\$dstW</code> .
<code>header</code>	<code>header("Content-type: image/png")</code> <code>header("Content-type: image/jpeg")</code> <code>header("Content-type: image/gif")</code>	Le operazioni per l'invio in output sono molto semplici, tuttavia è necessario cambiare il tipo MIME del file PHP, per impostazione predefinita infatti è su <code>text/html</code> , noi dobbiamo portarlo sul tipo della nostra immagine. Per fare questo ci serviamo della funzione

header() (serve quindi per specificare il tipo di output). Prima del richiamo della funzione di output è bene quindi utilizzare questa funzione nel modo specificato affianco.

imagepng	<code>imagePng(\$imageRef)</code>	Invia in output l'immagine \$imageRef nel formato PNG.
imagegif	<code>imageGif(\$imageRef[, \$path])</code>	Invia in output l'immagine \$imageRef come GIF, se si utilizza anche il secondo parametro, l'immagine sarà salvata e non inviata in output. Questa funzione, come tutte le funzioni collegate alle GIF, non è disponibile se si utilizza una versione 2.x delle GD.
imagejpeg	<code>imageJpeg(\$imageRef [, \$path [, \$qualità]])</code>	Specificando solo il primo parametro invia in output l'immagine \$imageRef in formato JPEG, specificando anche il secondo l'immagine non verrà inviata in output ma salvata con il nome indicato in \$path, il terzo parametro, sempre opzionale, ne definisce la qualità, si può inserire un valore compreso tra 0 (immagine scadente) e 100 (immagine ottimale). Se omesso viene usato 75 come valore predefinito. La qualità dell'immagine influirà ovviamente sulle dimensioni finali del file.
imagedestroy	<code>imageDestroy(\$imageRef)</code>	È possibile eliminare l'immagine dalla memoria prima che lo script sia terminato utilizzandola. L'utilizzo di questa funzione è superfluo se l'output dell'immagine è l'ultima operazione dello script in quanto l'interprete libererà automaticamente la memoria impiegata dal programma.

Abbiamo visto le principali funzioni GD per le immagini e stringhe di caratteri; vediamo ora il codice da usare ogni volta che si vuole mandare ad output una immagine (questo è il codice usato negli esempi delle funzioni elencate sopra):

```
<?php
// 1) crea una immagine 200*200
$img = imagecreate(200, 200);

// 2) alloca qualche colore: crea uno sfondo quadrato bianco
$white = imagecolorallocate($img, 255, 255, 255);

// 3) setta ora il colore da usare per il cerchio
$white = imagecolorallocate($img, 255, 0, 0);

// 4) disegna un cerchio rosso
imagearc($img, 100, 100, 150, 150, 0, 360, $white);

// 5) output dell'immagine nel browser
header("Content-type: image/png");
imagepng($img);

?>
```

Riporto anche il codice per la creazione di Thumbnails a partire da immagini specificate:

```
<?php
$img = "img.jpeg"; // percorso al file dell'immagine
$dest = "miniature/"; // directory di salvataggio delle miniature create

// dimensioni della miniatura da creare <BR> $thumbWidth = 60; // larghezza
$thumbHeight = 60; // altezza
// livello di compressione della miniatura
$thumbComp = 90;
```

```
// creazione dell'immagine della miniatura
$thumb = imagecreate($thumbWidth, $thumbHeight) or die("Impossibile creare
la miniatura");
// apertura dell'immagine originale
$src = imagecreatefromjpeg($img) or die ("Impossibile aprire l'immagine
originale");

// copio l'immagine originale in quella della miniatura ridimensionandola
imagecopyresized($thumb, $src, 0, 0, 0, 0, $thumbWidth, $thumbHeight,
imageSx($src), imageSy($src)) or die("Impossibile ridimensionare l'immagine");

// salvataggio miniatura
imagejpeg($thumb, $dest, $thumbComp) or die("Impossibile salvare la
miniatura");

?>
```